APPLE-AIDS

A collection of 12 extremely useful utilities
by ALLEN WYATT

Kill DOS • Disk Map • Undelete Files • Disk Editor • Text Dump • Base Convert

HOWARD W. SAMS
Developed by Advanced Operating Systems

LIMITED WARRANTY

For a period of ninety (90) days from the date of original purchase at retail, the warrantor, identified below, warrants this software to load and run as a basic program for the indicated microcomputer model, to be free from defects in material and workmanship and to be merchantable and suitable for its stated purpose for the period of this warranty. This warranty may not be enlarged except in writing, signed by warrantor. A Customer Registration Card is enclosed in this package. Please complete and return the Card to the warrantor within thirty (30) days. THE WARRANTOR EXPRESSLY DISCLAIMS ANY IMPLIED WARRANTY INCLUDING THE WARRANTY OF MERCHANTABILITY AND THE WARRANTY THAT THE SOFTWARE IS SUITABLE FOR ITS STATED PURPOSE AS OF THE DATE NINETY (90) DAYS FROM THE ORIGINAL PURCHASE OF THE SOFTWARE AT RETAIL.

•

In the event of defect, malfunction or failure of the software to conform with this warranty, the warrantor will repair or replace the software at no cost to you. For warranty service, you should return the software to the warrantor, Howard W. Sams & Co., Inc., 4300 W. 62nd Street, Indianapolis, Indiana 46268. Software received damaged as a result of shipping will require you to file a claim with the carrier. This warranty gives you specific legal rights and you may also have some other rights which vary from state to state.

•

THIS WARRANTY IS LIMITED SOLELY TO THE ABOVE AND THIS WARRANTY AND ANY WARRANTIES IMPLIED BY STATE LAW WILL APPLY ONLY FOR THE PERIOD SET FORTH. (SOME STATES DO NOT ALLOW LIMITATIONS ON HOW LONG AN IMPLIED WARRANTY LASTS, SO THE ABOVE LIMITATIONS MAY NOT APPLY TO YOU.) THE WARRANTOR WILL NOT BE LIABLE FOR ANY LOSS, DAMAGE, INCIDENTAL OR CONSEQUENTIAL DAMAGES OF ANY KIND, WHETHER BASED UPON WARRANTY, CONTRACT OR NEGLIGENCE, AND ARISING IN CONNECTION WITH THE SALE, USE OR REPAIR OF THE SOFTWARE. (SOME STATES DO NOT ALLOW THE EXCLUSION OR LIMITATION OF INCIDENTAL OR CONSEQUENTIAL DAMAGES, SO THE ABOVE LIMITATION OR EXCLUSION MAY NOT APPLY TO YOU.) UNLESS OTHERWISE CONTRARY TO STATE LAW GOVERNING THE PURCHASE, THE WARRANTOR'S LIABILITY SHALL NOT IN ANY CASE EXCEED THE CONTRACT PRICE FOR THE SOFTWARE CLAIMED TO BE DEFECTIVE OR UNSUITABLE.

•

WARNING: THE UNAUTHORIZED USE, REPRODUCTION OR DUPLICATION OF THIS MATERIAL, OR ITS PUBLIC PERFORMANCE OR DISPLAY, BY ANY MEANS IN ANY MEDIA FOR ANY PURPOSE, WHETHER IN WHOLE OR IN PART, IS STRICTLY PROHIBITED. VIOLATORS WILL BE SUBJECT TO ALL CIVIL AND CRIMINAL PENALTIES.

HOWARD W. SAMS & CO., INC.
4300 W. 62nd Street
Indianapolis, Indiana 46268

APPLE-AIDS

by Allen L. Wyatt

Copyright © 1981 ℗ ADVANCED OPERATING SYSTEMS

26874

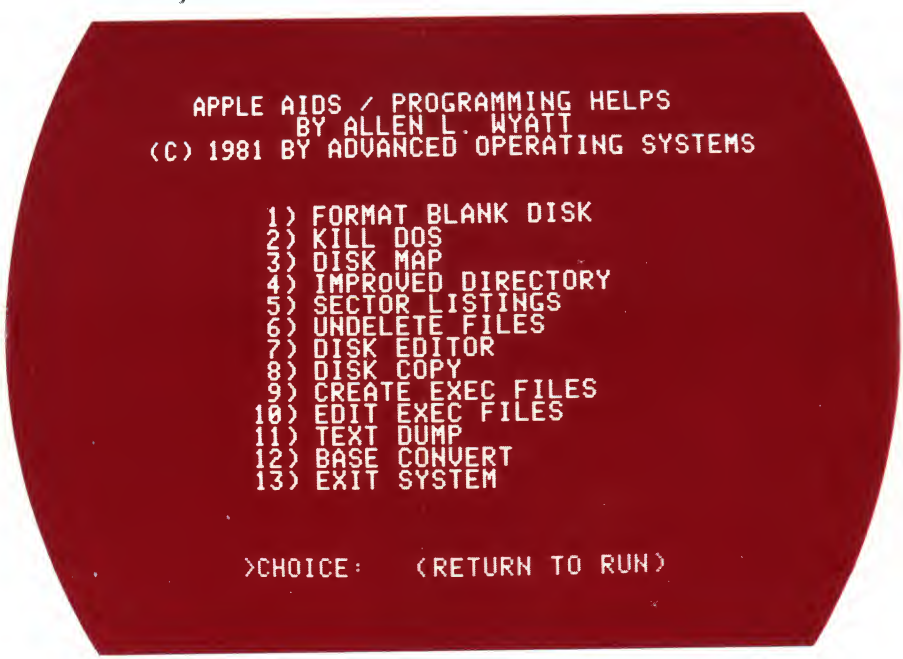
LOADING INSTRUCTIONS

Getting Started

In order to use APPLE-AIDS, you must have a 48K Apple II® or Apple II PLUS® computer. You must also have at least one disk drive.

APPLE-AIDS is written in Applesoft® and Machine Language. In order to use APPLE-AIDS properly, you must boot your system up using your APPLE-AIDS disk. You should execute the following steps:

1. Insert your APPLE-AIDS disk into drive 1.
2. Turn your computer power on, or type PR#s ("s" is the slot containing the controller card of the drive from which you wish to boot) and press RETURN .
3. You will see the APPLE-AIDS Main Program Menu appear on your CRT.



You are now ready to use any of the programs you see listed on your CRT. If the preceding three steps fail, check all of your electrical connections (including pc board contacts) and repeat the steps again.

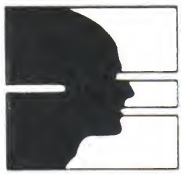
After you see the Main Program Menu, it would be a good idea to make a back-up copy of your APPLE-AIDS program disk. You can do this by choosing option 8 from the menu. Chapter 9 in this manual explains making back-up copies in more detail.

APPLE-AIDS will work with any Apple II® computer that has Applesoft in ROM® or a Language Card. If you have a standard Apple II®, with a Language Card, please do the following:

1. Insert a System Master diskette into your drive.
2. Boot up to this disk so that Applesoft® will be loaded into your Language Card.
3. Insert your APPLE-AIDS disk, type PR#6 and press RETURN .

If all else fails, contact your dealer. They can help you with most any problem you may encounter.

TABLE OF CONTENTS



APPLE-AIDS

- Introduction 7
- Format Blank Disk 9
- Kill DOS 11
- Disk Map 13
- Improved Directory 15
- Sector Listings 17
- Undelete Files 19
- Disk Editor 21
- Disk Copy 31
- Create Exec Files 33
- Edit Exec Files 35
- Text Dump 37
- Base Convert 39
- Exit System 41
- User Notes 43
- Glossary 44
- Appendix A—VTOC 47
- Appendix B—Diskette Directory 49
- Appendix C—Track Sector Lists 52
- Appendix D—How DOS Deletes A File 54
- Appendix E—Disk Potpourri 56
- Appendix F—Apple ASCII Chart 57
- Appendix G—ACSII Code Table 58
- Index 61

Chapter 1

INTRODUCTION

It has always been important for a personal computer user to be able to have access to "real" information. That information might be timely data, or it could be hard-to-find reference material. If there is a way to get such information cheaper or easier, the method of such acquisition generally is sought after.

Easy information retrieval and manipulation was the reason for this software package. At the time, there was no software package on the market that allowed the user to do all the things that needed to be done in software development. Thus, a set of tools was developed and refined to help in the development of other software products. These tools needed to be powerful, fast, friendly, and reliable.

This is the idea behind Apple-Aids. We think you will find them easy to use and helpful to you in your understanding and development of software. Before you start, however, we need to give a word of caution.

Please be extremely careful in your use of these utilities. Just as with all powerful tools, there is the ability to destroy as well as create. ALWAYS MAKE BACK-UP COPIES OF ALL WORK! This way, if something does go wrong, you are not left with a lot of ruined work. Make sure you understand EXACTLY what you intend to do, as well as what should happen, before you attempt to do it.

This manual has been written, for the most part, with the idea that the user would be using it on a DOS 3.3 system. If you are using it on 3.2, all operations will be the same with the exception that you will not be able to access anything above sector 12 ($C) because there are only 13 sectors on a DOS 3.2 initialized diskette.

As a final note, please read the Loading Instructions "GETTING STARTED" before attempting to use Apple-Aids. It may save you some time.

Chapter 2

FORMAT BLANK DISK

When you format a disk using the "INIT" command of DOS, your Apple writes a series of identifing marks on the diskette so that it can find its way around when it needs to. After doing this, the disk is left completely blank (for a short time). At this point, there are 35 tracks that have been written, each consisting of 16 sectors (or 13 sectors for DOS 3.2). This totals (by simple math), to 560 (or 455) blank sectors. Then "INIT" continues and marks off 64 (or 52) sectors for internal usage. Tracks Ø, 1, and 2 are used for an image of the Disk Operating System, and track 17 ($11) is used for the VTOC and the CATALOG. Then the "HELLO" program you have specified is linked and saved so that it runs when you boot DOS. However, all this leaves you 64 (or 52) sectors short of full disk usage.

This program will format your diskette in a slightly different manner. What happens is that the disk is marked into tracks and sectors, and no image of DOS is saved on the disk. However, track Ø and 17 ($11) are still "off limits" to a user. Track Ø is not accessable (except when booting or seeking directly, as with the Disk Editor), and track 17 ($11) is still required so that DOS can keep track of what you have saved on the disk.

To run the program, choose option 1 from the Main Program Menu. Then decide which Slot and Drive from which you wish to format, and follow the instructions.

An important thing to remember is that you will not be able to boot your system using your newly created data disk, since there is no DOS image on the disk. If you do attempt to boot to it, a message will be displayed on the screen notifying you of your error. In order to recover from this point, you will need to insert another system disk and type PR#6 to reboot the system again. Other than that, the disk will behave and appear as normal, except that the storage capacity has been increased by two tracks.

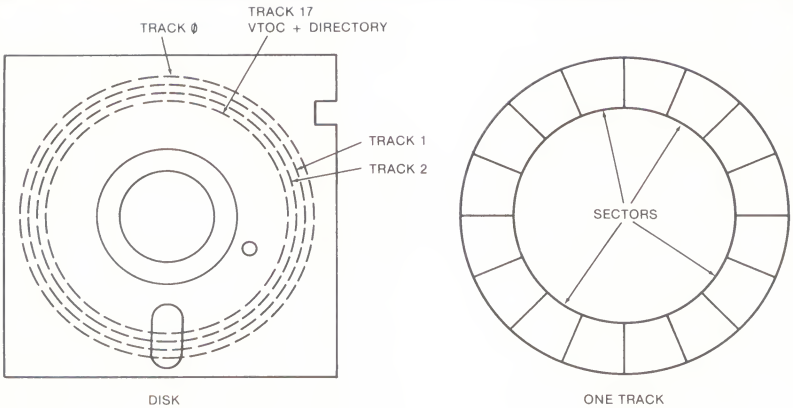


Fig. 1—Track and Sector locations on a disk.

Chapter 3

KILL DOS

KILL DOS will allow you to accomplish the same objective as FORMAT BLANK DISK, but in a slightly different manner. This program assumes that you already have a normally formatted diskette and that you have saved things to the disk already. In fact, you may have saved so much on the disk that you have found that you have run out of room to store some data you need.

Never fear! KILL DOS will allow you to take DOS off of your diskette without harming so much as a single byte of your valuable information. Again, this will "free-up" 32 sectors for your use, but you will not be able to boot your system using your disk (on the Apple, it is a pretty good trick to be able to boot without something with which to boot).

Once you have removed DOS, there are only two ways you can restore it. One is to re-initialize the diskette. The problem with this, though, is that you will loose all of the data you have on your diskette. You could also use "Master Create" (supplied by Apple), but if you have saved any information on tracks 1 or 2, you will still loose that data. So, make sure you want to KILL DOS before you actually do it.

To run this function, simply choose option 2 from the Main Program Menu, and determine on which Slot and Drive you wish to perform the operation. If you have more than one drive on your system, take care that you are "killing" the DOS on the correct drive.

As it is with FORMAT BLANK DISK, you will not be able to boot to your data disk any more. If you attempt to do so, you will be given an error message and returned to BASIC. All you need to do then is reboot to another disk with DOS.

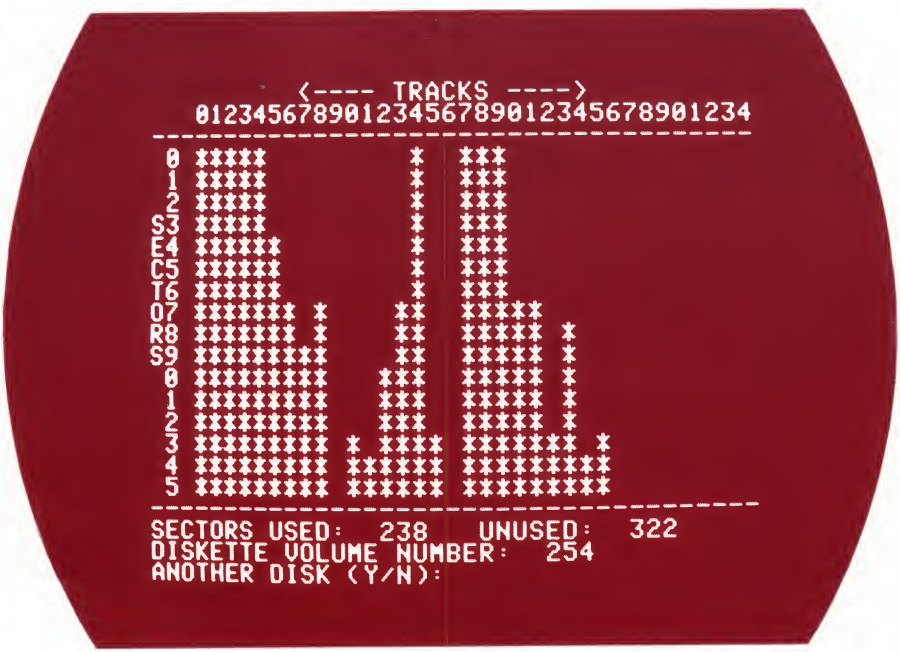
Chapter 4

DISK MAP

Many times, especially when doing highly specialized application software, you need to see exactly which sectors are in use or free on a given diskette. DISK MAP allows you to determine this information from any normally formatted diskette.

To use this utility, choose option 3 from the Main Program Menu. Then simply determine the Slot and Drive you wish to map and press RETURN when so prompted. Be patient, it takes a few seconds to map the disk completely.

Now you will see a graphic representation of the space allocation on the disk. "In use" sectors will be denoted with an asterisk (*). On this graph, tracks are represented on the horizontal axis, and sectors on the vertical axis. Hence, if you wish to determine if track



13 ($D), sector 15 ($F) is in use, you would locate the track across the top of the chart, and the sector along the side. If there is an asterisk (*) where the row and column meet, then the sector is in use.

DISK MAP also keeps count for you, and will let you know total sectors free as well as total sectors in use. This is handy to determine if you have enough room on a disk to transfer a file completely to the disk. These totals can be found at the bottom of the screen.

At the end of the mapping session, you have the option to map another disk. If you answer by pressing N , you will be returned to the Main Program Menu.

Chapter 5

IMPROVED DIRECTORY

How many times have you wished you knew exactly what was on the catalog of a disk? IMPROVED DIRECTORY will let you see this information.

To use IMPROVED DIRECTORY, choose option 4 from the Main Program Menu. As usual, you will be given a choice as to the Slot and Drive you wish to utilize. When you have entered the appropriate responses, simply press RETURN and you will be shown the file name (truncated at 10 characters), the file type (A=Applesoft, I=Integer, B=Binary, T=Text, S=S type, R=Relocatable file, ?=Who Knows), and the file code.

The file code is a one character notation of the status of the file. A file may either be unlocked, locked, or deleted. A blank, or no

mark, denotes that the file is unlocked. An asterisk (*) denotes the file is locked, and a dollar sign ($) denotes that the file is deleted.

This may appear a bit baffling at first, but DOS does not erase a file name from the directory simply because it has been deleted. For more information on the deletion process, see the reference sections at the end of this manual.

Besides the file code, if the file type is 'B' (Binary), you will be shown the beginning address and file length of the program. Both of these numbers are in decimal notation. This can be very helpful, since this information is not shown with normal "CATALOG" function of DOS.

Chapter 6

SECTOR LISTINGS

Sector listings are very handy if you wish to pin-point (exactly) where a file resides on disk. This program provides you precisely with what it suggests: a list of the tracks and sectors used by the files on a disk.

In order to use this utility, choose option 5 from the Main Program Menu. When you answer the usual questions as to Slot and Drive, and then press RETURN , you will be shown a series of numbers in the format (TT-SS) In other words, the track is shown first, and then the sector, so that if you see the numbers (17-15), part of the file you are examining is contained on track 17, sector 15.

The numbers shown in inverse (black on white) denotes that the particular sector shown is used for the track-sector list of the file being examined. All other sectors are shown in normal mode (white on black.)

Also, it is important to remember that all information shown is represented in decimal notation.

Chapter 7

UNDELETE FILES

I don't know about you, but there are many times when I have deleted files that I didn't want to, but once I hit RETURN , it was too late. This usually has happended late at night when I've been up for 43 hours straight and am having trouble seeing the monitor, let alone hitting the right keys.

Be that as it may, I am left with no way to recover a file I just deleted, right?

Wrong! All you need to do is to use UNDELETE FILES to salvage that lost program, and it will "magically" reappear in complete form. If you wish to learn how this is done (in detail), turn to the reference sections at the end of this manual.

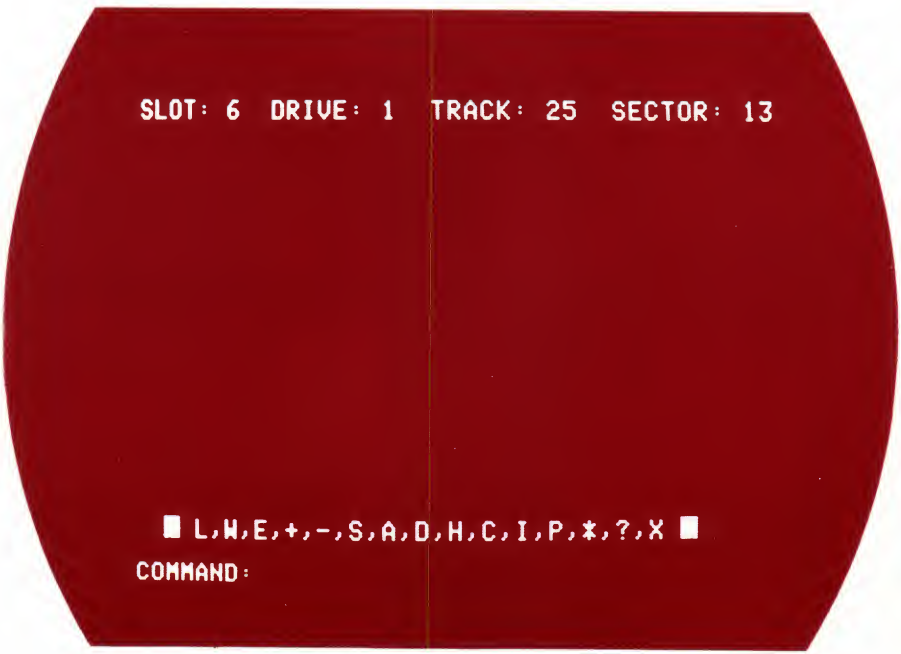
In order to use the program, choose option 6 from the Main Program Menu. Again, you are given the choice of Slot and Drive. After entering the proper choices, press RETURN to begin the review. As each previously deleted program is located in the directory, you will be shown the file name and given a choice whether to "undelete" the file, or not. If you answer "N", the next file will be searched for. If you answer "Y", the program sets relative byte Ø of the Directory back to its original value, and puts the track bit map back to its "pre-deleted" values. Now the file is resurrected and all is as before.

There are a few pointers to remember, however. If you have deleted a program and later decide to undelete it, this will only be meaningful if you have not saved anything else out to the disk since you last deleted the file. If you did, it is very likely that DOS, thinking you meant it when you deleted the file, has written your new file into part, if not all, of the space previously occupied by the deleted file. If you try to undelete under these circumstances, chances are you will mess up the directory; and that will mean extensive patch work later on, a blown directory, or garbaged files. In any case, it is a good practice to use UNDELETE FILES only immediately after having deleted the file.

Chapter 8

DISK EDITOR

This is the single most used program in our library. DISK EDITOR is a powerful utility that will allow you to load, inspect, alter, and write any sector of any normally formatted disk. To use DISK EDITOR, choose option 7 from the Main Program Menu and you will see the following:



To avoid possible inadvertant damage to your Apple-Aids disk, we suggest that you remove it from the drive at this time. You may then insert any disk that you wish to edit.

All of the DISK EDITOR commands shown in the command line are single-key commands. You do NOT need to press RETURN The commands and their subsequent actions are:

L—This command will LOAD any sector you designate into the editor buffer and then display it on the screen in the current display mode (for display modes, see commands A, D, and H).

W—This is the opposite of load, i.e., this command will WRITE the sector currently in the editor buffer to any place you designate on the disk.

E—EDIT. This will allow you to change any or all of the information displayed on the screen. These changes are made in the editor buffer and can then be written back out to the disk. The editing begins at whatever byte you specify, and you have the choice of several edit types. The edit types are:

D—Edit will be a one-byte DECIMAL change.

H—Edit will be a one-byte HEXADECIMAL change.

A—Edit will be an ASCII string change starting with the specified byte. For instance, with this type of edit you can enter a word as the change, and it will be inserted beginning with the starting byte you specified.

E—Edit will be an ENHANCED ASCII string change. This is the same as an ASCII change, except that the character values have the high byte on (ASCII value + $8Ø).

F—Edit mode will be "FILL". You will be asked for the ending byte and the type of fill (decimal or hex), and then the value with which to fill. All bytes between the limits you have set will then be changed to the desired value.

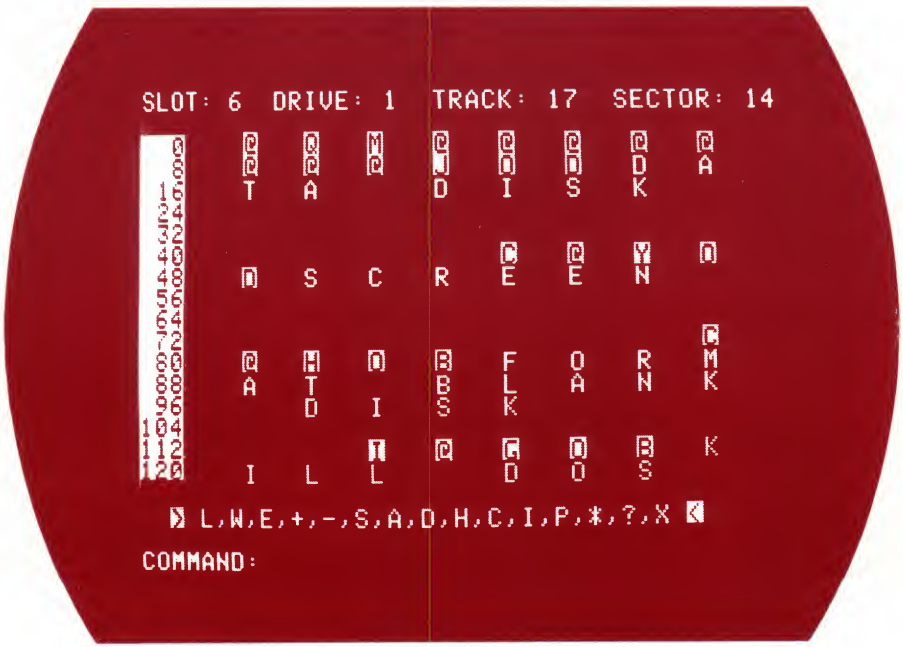
All edits are done immediately and displayed in the current display mode.

+—This command will allow you to INCREMENT the sector pointer by one, LOAD and DISPLAY in the current mode. Thus, if you are currently viewing track 3, sector 4 on the display, upon pressing + track 3, sector 5 will be loaded and displayed. Also, track borders are automatically bridged, so that if you are on track 34 ($22), sector 15 ($F) and you press +, track Ø sector Ø will be loaded and displayed.

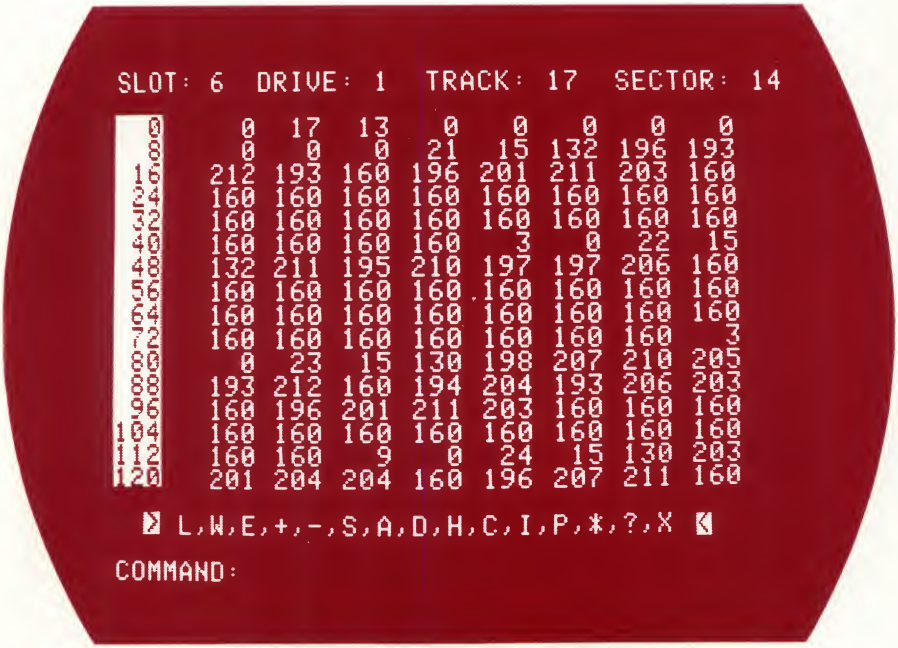
–—The '–' command works the same as the '+' command, but in reverse. Pressing – will DECREMENT the sector pointer by one, LOAD and DISPLAY in the current mode. Track borders are automatically bridged here, also.

S—This command SWITCHES the part of the sector being viewed on the CRT. Each page of display can accommodate 128 ($8Ø) bytes of data. The first page display comprises bytes Ø-127 ($Ø-$7F). The second contains bytes 128-255 ($8Ø-$FF). Pressing S will toggle between the two halves of the sector. The current display mode (A, D, or H) is retained.

A—This displays the current page of the editor buffer in ASCII representation. ASCII characters in the range Ø-$3F and $8Ø-$9F are shown in inverse (black on white) print. Characters in the range $4Ø-$7F are flashing, and characters between $AØ and $FF are shown in normal print. This display mode is maintained until another display mode is chosen.



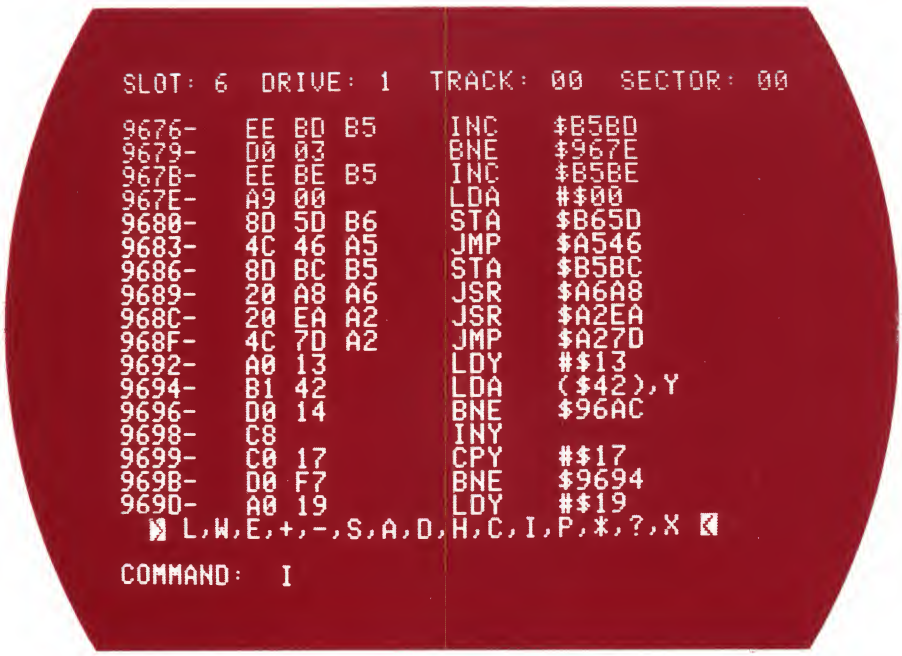
D—This displays the current page of the editor buffer in DECIMAL notation. This display mode is maintained until another display mode is chosen.



H—This displays the current page of the editor buffer in HEXADECIMAL notation. This display mode is maintained until another display mode is chosen.

C—This command will CONDENSE the data and show the contents of the entire sector in hex and ASCII format on the CRT at one time. This is for those who may have a need to see the sector displayed as such. This is a viewing mode only. All edits must be done from any of the other display modes (A, D, or H). In order to return to the current display mode, simply press any key.

I—This command will DISSASSEMBLE the current sector buffer. Standard 6502 mnemonics are used and displayed. You can use CTRL-S as a toggle to start/stop the display. To get back to a normal display, simply press any of the display mode commands (A, D, or H).



P—This is a PRINTER TOGGLE. If you press P , you will be asked for your printer slot. All information and output will then be directed to your printer. Pressing P again will turn your printer off.

*****—This command will allow you to change the currently logged Slot and Drive on which you are working . A valid slot must be given. Using this command will NOT destroy or clear the buffer.

?—This will show you a LIST OF THE COMMANDS AND EDIT TYPES available. To get back to normal display, simply press any of the display mode commands (A, D, or H).

X—Use this to EXIT the program back to the Main Program Menu.

A word must be said about general editor format. All byte markers are shown to the left side of the screen. These refer to the first byte of the row. There are eight bytes per row, sixteen rows per page, two pages per sector. This is not true in the Condensed viewing mode. In that mode, there are 12 ($ØC) bytes per row, and the entire sector is shown on one page.

Both display and input is determined by the display mode you are in currently. If you are in Hex mode, all display and input is given and expected in Hexadecimal notation. If you are in Decimal or ASCII notation, all display and input are given and expected in decimal notation. The only exception to this is when you are in Edit mode and you are specifying the type of edit to be made. In such cases, you can make a Hexadecimal change or fill while in Decimal or ASCII display modes. You can also make a Decimal or ASCII change or fill while in Hex display mode.

You will notice status markers at the top of the editing screen. These show you on which Slot and Drive you are functioning, and which track and sector is currently in the buffer. If you wish to write a sector back out to the same place from which you loaded it, just enter the same track and sector that you see noted by the indicators.

If you wish to write a sector from one disk to another, simply Load the sector from one disk, then change the Slot and Drive parameters (Use the "*" command). Now write the buffer to any place on the other disk you wish. If you are only using one disk drive, you will need to switch disks physically in your drive to accomplish this same task.

There are no operational differences between the DOS 3.3 and the DOS 3.2 versions of DISK EDITOR. There is one application difference, however, that should be noted.

If you try to read a DOS 3.2 sector that has never had anything written to it, a drive error will be generated. This is due to the fact

that DOS 3.2 leaves a sector COMPLETELY blank until written to. DOS 3.3 does not have this problem, because when DOS 3.3 initializes a disk, it writes zeros to each byte of every sector on the disk, thus no errors are generated when a "blank" 3.3 formatted sector is read. This is only a minor inconvenience of the 3.2 operating system and will in no way affect the other operations of the Editor. You can still write to any of these "unreadable" sectors, and then you will have no problem later reading from them.

Please see the reference sections at the end of this manual for a disk map that shows where certain standard items are located on the disk.

Chapter 9

DISK COPY

This program will copy a disk to another disk. Either one or two disk drives can be used, and there is no need to format the disk being copied onto. This program does that automatically.

All you need to do is choose option 8 from the Main Program Menu. Then determine in which Slot and Drive the original (master) disk will be, and in which Slot and Drive the copy (blank) disk will be. When you press RETURN , the disk denoted as the copy (blank) will be formatted and copied to track-by-track.

Notice that you are kept constantly informed regarding the status of the copy procedure. At the successful conclusion of the copy, you will be given the chance to make another copy (parameters can be reset at this time), or to end. If you end, you will be returned to the Main Program Menu.

You can abort a copy session at any time by pressing the ESC key. If you do so, the copy process will be halted at the end of the current write cycle and you will be notified that the copy has been aborted. You will also be notified of the last track written to the copy disk.

Let us note here the purpose of copy programs. This copy program is here for your convenience and use. It is intended for you to use in order to make personal back-up copies of your programs. It is NOT meant for you to use to try to copy any copyrighted software for other than personal use. If you do use it for such, you become what is referred to as a "Software Pirate". As such, you are stepping outside the law and your legal rights. If you find it necessary to do so, please be assured that AOS will not support pirated copies of APPLE-AIDS, nor will they support the pirate. If you have any questions about our policy on such matters, please call or write for information.

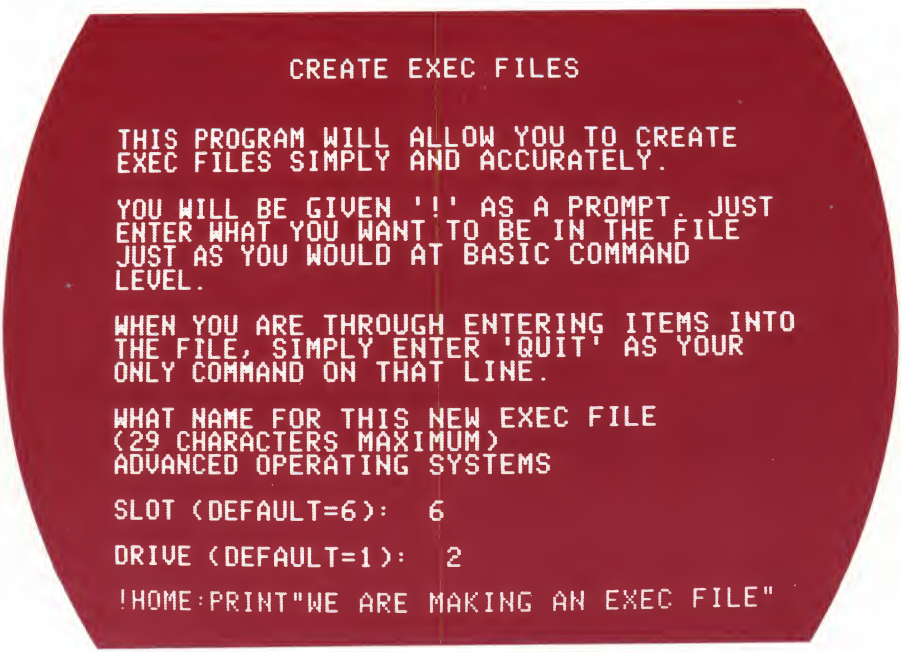
Chapter 10

CREATE EXEC FILES

Creating Exec files can be a headache. This utility will allow you to put together a series of commands into a text file to be executed later. With a little practice, creating Exec files can become fairly easy to do.

In order to use this utility, choose option 9 from the Main Program Menu. Now you will be asked for the name of the file. Enter any valid file name. It can contain any characters except a comma (,) or colon (:), and it cannot begin with a number. It also cannot be over 29 characters in length.

Next, enter the Slot and Drive on which you will be working. Make sure you have the disk on which you want the Exec file saved in the disk you specify.



Once you have done all of the above, you will see an exclamation mark (!) as a prompt. You can now enter any commands just as you would at command level, and they will be saved out to the disk. Commas, colons, and quote marks are allowed here, so you can type just about anything you want.

In order to end a creating session, type the word "QUIT" as your command. It must be the only word on the line. You will then be returned to the Main Program Menu.

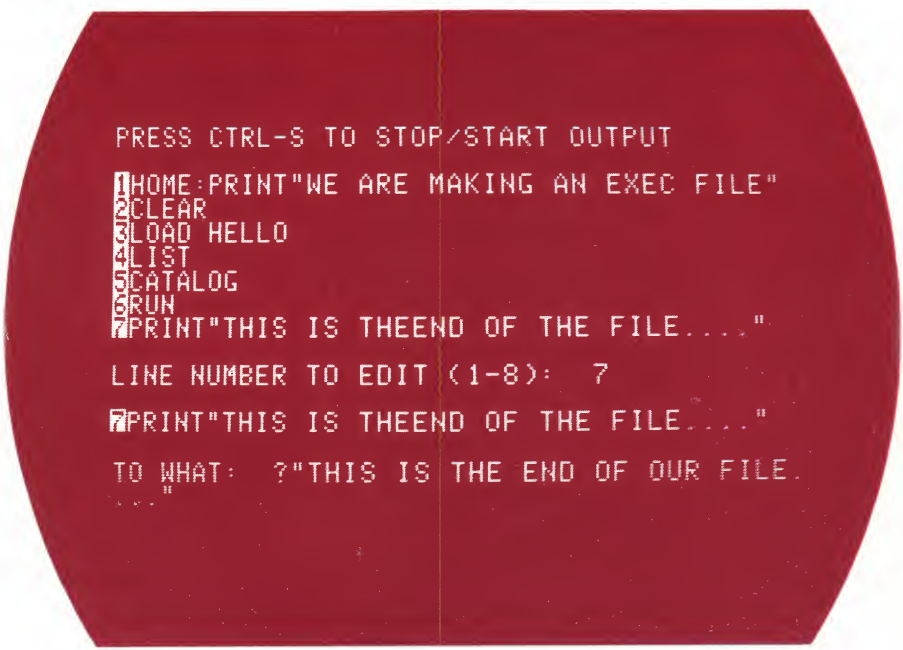
Chapter 11

EDIT EXEC FILES

Most of the time, it is impossible to get an Exec file correct the first time through. It can be very frustrating and tedious to retype the entire file just to correct one mistake. On the other hand, if you do happen to get it right the first time, it could be considered a miracle and reason for celebration. This is particularly true as your Exec files increase in size.

EDIT EXEC FILES will allow you to load, inspect, change, and rewrite any Exec file on a disk. All you need to do is provide the name of the file and any changes you want to make.

To use this program, all you have to do is choose option 1Ø from the Main Program Menu. Next you must enter the name of the file to be edited and the Slot and Drive location. The file will then be read into memory and shown on the screen.



This is a line-oriented editor, so if you are changing a line, you must retype it entirely. You can add lines to the end of the file, or you can delete lines from any place within the file. To do this choose the line number you wish to edit. In order to add lines to the end of the file, choose a line number one higher than the last one in the file. For example, if there are 1Ø lines in your file, enter "11" as the line to edit. You will be notified that you are entering a new line. Then just enter your commands as normal.

In order to end editing, enter "QUIT" as your command, just as in CREATING EXEC FILES. Your edited file will then be saved out to the disk you specified. After this, you will be returned to the Main Program Menu.

Chapter 12

TEXT DUMP

This is a machine-code program that will allow you to dump to printer whatever is on the screen.

To use this program, choose option 11 from the Main Program Menu. Then supply your printer slot. Now, anytime that you press Ctrl-P , whatever is on the screen will go to the printer.

This will only work when the Apple is waiting for input from the user. Usually this is anytime you can see the blinking cursor.

This feature will stay in effect until you do one of the following:

1. Press RESET
2. Type PR#6
3. Load another machine code program that writes over Page 3 of memory.

Once this routine has been implemented, you will be returned to the Main Program Menu.

Chapter 13

BASE CONVERT

BASE CONVERT works on the premise that it is easier to let the computer do what it was intended to do than to do it yourself. This program works with binary, octal, decimal, and hexadecimal numbers. It will convert a number from any of the above bases to all of the bases.

All you need to do is to enter the base code and then the number in the format "CN". This "C" is the base code, and the "N" is the number to be converted.

For example, if you wish to convert the decimal number 3456, type "D3456" and then press RETURN . The computer will respond with:

```
                BASE CONVERT

THIS PROGRAM WILL CONVERT A NUMBER FROM
BINARY, OCTAL, DECIMAL OR HEXADECIMAL
TO BINARY, OCTAL, DECIMAL & HEXADECIMAL.
JUST ENTER B,O,D, OR H AND THEN THE
NUMBER.  FOR EXAMPLE, DECIMAL 123 WOULD
BE ENTERED AS 'D123'.

ENTER 'QUIT' AS THE NUMBER TO END THIS
PROGRAM.

(B,O,D,H) NUMBER:  D3456

BINARY: 110110000000
OCTAL: 6600
DECIMAL: 3456
HEX: D80

(B,O,D,H) NUMBER:
```

Numbers can be converted from any of the other bases simply by using a different base code: "B" for Binary, "O" for Octal, "D" for Decimal, or "H" for Hexadecimal.

In order to exit the program, simply enter the word "QUIT" when prompted for input, and you will be returned to the Main Program Menu.

Chapter 14

EXIT SYSTEM

This is the proper way to exit APPLE-AIDS. It should be noted here that APPLE-AIDS uses error-handling routines that prevent certain things from happening.

Pressing Ctrl-C may result in the system hanging indefinitely.

Pressing RESET will result in the system rebooting.

Chapter 15

USER NOTES

1. All Advanced Operating Systems programs are extensively tested and debugged. However, you are the final judge. If you feel that you have discovered something that should not be happening (i.e. a "bug"), then WRITE DOWN the conditions and steps that led you to the discovery and apparent problem. Then send us a note and we will figure out what happened.

2. All Advanced Operating Systems programs are covered by our limited warranty. Specific warranty information is contained elsewhere in this manual.

3. The only way that our guarantee will apply is if you fill out and return the user registration card contained in this package. It lets us know that you are indeed the owner of this software package.

GLOSSARY

ASCII: **A**merican **S**tandard **C**ode for **I**nformation **I**nterchange. A standard 8-bit (1 byte) information code used with most computers and data terminals.

BASIC: **B**eginners **A**ll-purpose **S**ymbolic **I**nstruction **C**ode. A widely used computer language. Particularly popular on micro computers.

BINARY: A numerical system based on the powers of 2. The only digits used in binary are "Ø" and "1". See also BIT.

BIT: A binary digit. This can either be a "Ø" or a "1".

BOOT: Short for "Boot-strap". Refers to the process of bringing DOS and a disk system on line. On the Apple, this includes reading DOS from tracks Ø, 1, and 2 and executing the "Hello" program (if any).

BYTE: Eight bits. A byte may have any value of Ø through 255.

CATALOG: DOS command that prints the directory of files to the current output device. Also interchangeable with DIRECTORY when refering to a physical section of a disk.

CONTROL CHARACTERS: Any ASCII character in the value range of $Ø1 through $1F. Most of these characters can be typed from the Apple keyboard, but they will not appear on the CRT. They are used to control certain functions such as line feed, carriage return, and bell output.

CPU: **C**entral **P**rocessing **U**nit. The primary processor for a computer. It contains the arithmetic units, logic units, and the status registers for the computer. In the Apple II, the CPU is the 65Ø2.

CRT: **C**athode **R**ay **T**ube. A display monitor frequently used with computers. A television set is a type of CRT.

DECIMAL: A numerical system based on the Powers of 1Ø. Uses the digits "Ø" through "9".

DIRECTORY: Normally contained on track 17 ($11), sectors 1-15 ($Ø1-$ØF) of a diskette. Contains information that allows DOS to locate and check the status of individual files.

DISASSEMBLE: A term used to describe the process of translating machine code to assembly language.

DOS: Disk **O**perating **S**ystem. The set of commands and instructions that permit the orderly flow of data and programs between a mass storage peripheral (a disk subsystem) and the CPU.

EPROM: Erasable **P**rogrammable **R**ead **O**nly **M**emory.

EXEC FILE: A text file containing commands or program lines to be executed by the Apple. The commands stored appear just as they would if they were typed at command level from the keyboard.

FIRMWARE: Those programs and instructions that reside in ROM or PROM for use on a computer system.

HARDWARE: The physical circuits and materials that make up a computer or its peripherals.

HEXADECIMAL: A numerical system based on the powers of 16. Hexadecimal is frequently used in computers as an alternative to BINARY because of its relative ease of understanding. The digits "Ø" through "9" and the letters "A" through "F" are used. For example, "2E" means 14 ones and 2 sixteens, or 46 (decimal).

K: Kilobyte. A unit of measure refering to 1Ø24 Bytes.

MONITOR: See CRT.

MONITOR: The series of instructions written in machine code that take first control of the computer when it is powered up. The monitor on the Apple resides in ROM.

PERIPHERAL: A device that can be attached to a computer to enhance its inherent abilities. Some examples are printers and disk drives. Most peripherals are considered Input/Output devices.

POWER-UP: The process of applying power to a computer to start it up. In plain English, this involves flipping the switch to supply power to the computer.

PROM: Programmable **R**ead **O**nly **M**emory.

RAM: Random **A**ccess **M**emory. The main storage and work area of a computer. RAM is usually measured in K.

ROM: **R**ead **O**nly **M**emory.

SECTOR: A storage area used by DOS to store data on a disk. Each Track on a disk is divided into 16 (DOS 3.3) or 13 (DOS 3.2) sectors. Each sector can store 256 bytes of data.

SOFTWARE: Computer programs that can be loaded into RAM and contain a series of executable steps for the computer to process.

TOGGLE: A "switch" used to turn something on and off. In computers, this can be a command or a memory location. For example, accessing location $CØ3Ø will produce an audible click on the speaker of the Apple.

TRACK: A sequence of binary cells arranged on a disk so as to enable data to be stored and retrieved in a logical manner. There are 35 tracks in standard Apple DOS. See also SECTOR.

TRACK BIT MAP: The section of VTOC that tells DOS which tracks and sectors are free.

TRACK/SECTOR LIST: A sector on the disk which points to the individual tracks and sectors occupied by any given file on that disk. Each file has its own track/sector list.

UTILITY: A program developed for the express purpose of aiding the development or diagnosis of other programs, data files, or disk files.

VTOC: **V**olume **T**able **O**f **C**ontents. Track 17 ($11), sector Ø on a normally formatted Apple diskette. The VTOC contains information that tells DOS which tracks and sectors are in use and where the first directory sector is located.

Appendix A

VTOC
(Volume Table of Contents)

The VTOC, located at Track $11, Sector $Ø, is used by DOS to locate the first directory sector and to tell which tracks and sectors on the disk are free for use. It also contains certain other controlling information that is essential to the proper operation of DOS.

VTOC FORMAT

BYTE	PURPOSE/CONTENT
$ØØ	Not used
$Ø1	Track number of first directory track (usually $11)
$Ø2	Sector number of first directory sector (usually $ØF)
$Ø3	Release number of DOS
$Ø4-$Ø5	Not used
$Ø6	Diskette volume number ($Ø1-$FE)
$Ø7-$26	Not used
$27	Maximum track/sector pairs per Track/Sector List sector (normally $7A)
$28-$2F	Not used
$3Ø	Last track where sectors were used
$31	Track allocation direction (+1 or −1)
$32-$33	Not used
$34	Tracks per diskette
$35	Sectors per track
$36-$37	Bytes per sector (L-H)
$38-$3B	Track Ø bit map
$3C-$3F	Track 1 bit map
$4Ø-$43	Track 2 bit map
$44-$47	Track 3 bit map
$48-$4B	Track 4 bit map
$4C-$4F	Track 5 bit map
$5Ø-$BB	Bit maps for tracks 6 through $2Ø
$BC-$BF	Track $21 bit map
$CØ-$C3	Track $22 bit map
$C4-$FF	Not presently used

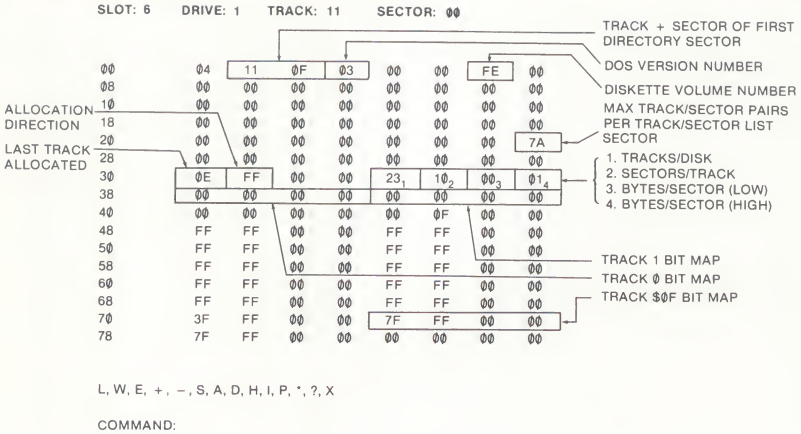


Fig. 2—First part of VTOC Sector

NOTES

1. In theory the diskette volume number in byte $06 can only be of value $01 through $FE. In practice, with special programs, you could use $0 or $FF, but using volume $00 will not allow linking and running of a "Hello" program. In other words, booting up a disk with a volume number of "0" will not run a "Hello" program. You will be returned to BASIC when DOS has been loaded and the DOS hooks established.

2. The volume number is also stored on each track and sector header. Changing this byte alone will NOT change the volume of a diskette.

3. Track bit maps are allocated 4-bytes per track, a total of 140 bytes. The second 2 bytes are not used. The first 2 bytes are used in the following manner:

	BYTE 1								BYTE 2							
FLAGS →	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
SECTOR →	F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0

A "0" in any position denotes that the sector is in use; a "1" denotes that the sector is free.

Appendix B

THE DISKETTE DIRECTORY

Every normally formatted Apple diskette has a directory. This is essential so that DOS can find any file that you or a program asks for. The directory is laid out in a very systematic manner, and can be viewed and adjusted by a user with a little practice.

The directory starts at Track $11, Sector $0F. Each directory sector can contain 7 entries. This first directory sector is linked to the second directory sector at location $11-$0E), which is linked to the sector below it, and so on. The last directory sector is located at ($11-$01). This is a total of 15 sectors. At 7 entries per sector, there is a 105 file directory entry maximum on a normal diskette.

Each directory sector is configured in the same manner, so we will describe just one sector here.

DIRECTORY SECTOR FORMAT

BYTE	PURPOSE/CONTENTS
$00	Not Used
$01	Track for next catalog sector
$02	Sector for next catalog sector
$03-$0A	Not Used
$0B-$2D	First directory entry
$2E-$50	Second directory entry
$51-$73	Third directory entry
$74-$96	Fourth directory entry
$97-$B9	Fifth directory entry
$BA-$DC	Sixth directory entry
$DD-$FF	Seventh directory entry

NOTE: Bytes $01 and $02 will be set to 0, if the particular sector is the last directory sector.

SLOT: 6 DRIVE: 1 TRACK: 11 SECTOR: ØE

TRACK/SECTOR OF NEXT DIRECTORY SECTOR

TRACK # OF FILE T/S LIST

SECTOR NUMBER OF FILE T/S LIST

FILE TYPE

ØØ	ØØ	11	ØD	ØØ	ØØ	ØØ	ØØ	ØØ	
Ø8	ØØ	ØØ	ØØ	13	ØF	82	C1	DØ	
1Ø	DØ	CC	C5	AØ	C1	C9	C4	D3	
18	AØ	AØ	AØ	AØ	AØ	AØ	AØ	AØ	DIRECTORY ENTRY #1
2Ø	AØ	AØ	AØ	AØ	AØ	AØ	AØ	AØ	
28	AØ	AØ	AØ	AØ	Ø6	ØØ	14	ØF	SECTOR COUNT FOR FILE
3Ø	84	C3	CF	C4	C5	AØ	AØ	AØ	
38	AØ	AØ	AØ	AØ	AØ	AØ	AØ	AØ	DIRECTORY ENTRY #2
4Ø	AØ	AØ	AØ	AØ	AØ	AØ	AØ	AØ	
48	AØ	AØ	AØ	AØ	AØ	AØ	AØ	Ø2	
5Ø	ØØ	15	ØF	84	C4	C1	D4	C1	
58	AØ	C4	C9	D3	CB	AØ	AØ	AØ	DIRECTORY ENTRY #3
6Ø	AØ	AØ	AØ	AØ	AØ	AØ	AØ	AØ	
68	AØ	AØ	AØ	AØ	AØ	AØ	AØ	AØ	
7Ø	AØ	AØ	Ø3	ØØ	16	ØF	82	C6	
78	CF	D2	CD	C1	D4	AØ	C2	CC	DIRECTORY ENTRY #4

L, W, E, +, −, S, A, D, H, I, P, *, ?, X

COMMAND:

a) Hex.

SLOT: 6 DRIVE: 1 TRACK: 17 SECTOR: 14

Ø	@	Q	M	@	@	@	@	@
8	@	@	@	S	O	B	A	P
16	P	L	E		A	I	D	S
24								
32								
4Ø					F	@	T	O
48	D	C	O	D	E			
56								
64								
72								B
8Ø	@	U	O	D	D	A	T	A
88		D	I	S	K			
96								
1Ø4								
112			C	@	V	O	B	F
12Ø	O	R	M	A	T		B	L

L, W, E, +, −, S, A, D, H, I, P, *, ?, X

COMMAND:

b) ASCII.

Fig. 3—Sample Directory Sector

Each directory entry above is laid out in the following format.

DIRECTORY ENTRY FORMAT

BYTE	PURPOSE/CONTENTS
$00	Track of first track/sector list for the file
$01	Sector of first track/sector list for the file
$02	File type and lock/unlock notation (see below)
$03-$20	File name (30 characters)
$21-$22	Number of sectors used by file

NOTES

1. All of the byte references listed previously are RELATIVE. This means that if you are refering to the fourth directory entry, you would add the above byte locations to the entry offset. The fourth directory entry begins at byte $74 of the directory sector. The file type would be noted at byte $76 ($74+$02).

2. If a file is deleted, bytes $00 and $20 are affected. See the section on deleting files.

3. Byte $02 may have any of the following values:

LOCKED	UNLOCKED	FILE TYPE
$00	$80	Text
$01	$81	Integer
$02	$82	Applesoft
$03	$84	Binary
$08	$88	S type file
$10	$90	Relocatable object file
$20	$A0	A type file
$40	$B0	B type file

4. Even though 2 bytes are used to notate the number of sectors used by a file, only the low-order byte ($21) is used by the CATALOG function to show the number of sectors used by the file.

Appendix C

TRACK SECTOR LISTS

(Or, Where Do We Go From Here?)

Each file on a disk has a Track Sector List. This is a special sector on the disk that allows DOS to know which sectors come next for an individual file. In different computer systems, there are different methods of linking file sectors together. Some use pointers from one sector to the following sector, much the same way as Apple DOS links directory sectors. Others keep special lists for each file. That is what a track sector list contains. It is just a lot of numbers that point to where a file is physically located on a disk.

The following is the format for a Track Sector List sector.

TRACK/SECTOR LIST FORMAT

BYTE	CONTENTS/PURPOSE
$00	Not used
$01	Track number pointer to next T/S List sector ("0" if no link)
$02	Sector number pointer to next T/S List sector ("0", if no link)
$03-$04	Not used
$05-$06	Sector offset for first sector in list
$07-$0B	Not used
$0C-$0D	Track and sector of next file sector
$0E-$0F	Track and sector of next file sector
$10-$FF	Track and sector pairs (up to 120 more)

NOTE: The sector offset in bytes $05-$06 is used as a check to make sure that the entire file is being read correctly. If another Track/Sector List sector is needed, the link would be noted in bytes $01-$02. Since the first T/S pair noted in this second sector would actually be the 123rd sector of the file, this information would be stored in the offset bytes.

SLOT: 6 DRIVE: 1 TRACK: 1C SECTOR: ØF

POINTER TO NEXT TRACK AND SECTOR OF NEXT TRACK/SECTOR LIST SECTOR

SECTOR OFFSETT

TRACK/SECTOR PAIRS

ØØ	ØØ	ØØ	ØØ	ØØ	ØØ	ØØ	ØØ	ØØ
Ø8	ØØ	ØØ	ØØ	ØØ	1C	ØE	1C	ØD
1Ø	1C	ØC	1C	ØB	1C	ØA	1C	Ø9
18	1C	Ø8	1C	Ø7	1C	Ø6	1C	Ø5
2Ø	1C	Ø4	1C	Ø3	1C	Ø2	1C	Ø1
28	1C	ØØ	Ø3	ØF	Ø3	ØE	Ø3	ØD
3Ø	Ø3	ØC	Ø3	ØB	Ø3	ØA	Ø3	Ø9
38	Ø3	Ø8	Ø3	Ø7	Ø3	Ø6	Ø3	Ø5
4Ø	Ø3	Ø4	ØØ	ØØ	ØØ	ØØ	ØØ	ØØ
48	ØØ	ØØ	ØØ	ØØ	ØØ	ØØ	ØØ	ØØ
5Ø	ØØ	ØØ	ØØ	ØØ	ØØ	ØØ	ØØ	ØØ
58	ØØ	ØØ	ØØ	ØØ	ØØ	ØØ	ØØ	ØØ
6Ø	ØØ	ØØ	ØØ	ØØ	ØØ	ØØ	ØØ	ØØ
68	ØØ	ØØ	ØØ	ØØ	ØØ	ØØ	ØØ	ØØ
7Ø	ØØ	ØØ	ØØ	ØØ	ØØ	ØØ	ØØ	ØØ
78	ØØ	ØØ	ØØ	ØØ	ØØ	ØØ	ØØ	ØØ

L, W, E, +, −, S, A, D, H, I, P, *, ?, X

COMMAND:

Fig. 4—Sample Track/Sector List Sector.

Appendix D

HOW DOS DELETES A FILE

(And How We Can Undo It)

When a user or a program issues a DELETE command, the disk file is not completely erased from the disk. In a certain sense, it is never fully taken off of the disk unless it is written over or the disk is initialized again.

First of all, DOS, upon receipt of the DELETE command, will mark the directory entry for the file by moving relative byte Ø to relative byte 32 ($2Ø) and setting relative byte Ø to the value 255 ($FF). This is shown in Fig. 3 on page 50. Now the file will not appear when you ask for a "CATALOG" of the disk.

But this is not all, for now DOS updates the track bit map (VTOC) for any tracks on which the file may have been saved. Now, for all practical purposes, the file is gone, because next time you save something, it will be written into the space previously occupied by the file you just deleted.

The actual physical sectors have not been erased unless you write over them or initialize the disk. If you save another file to the disk, then all of the pointers to the old file location will be removed or erased, at this point.

As long as the file pointers have not been removed, the file (usually) is still recoverable. Using a program like UNDELETE FILES accomplishes this goal. We said usually, because, in theory, all that is needed is to restore relative byte $Ø to its "pre-deletion" value (remember, this value is stored in relative byte $2Ø). Then the track bit maps need to be adjusted to show the file sectors in use again. Once this is done, the file is effectively restored.

The problems arise if disk usage has occured since the file was first deleted. When you increase the size of an existing file, or save a new file to disk, chances are that it will be written into part, if not all, of the space previously occupied by the deleted file. If you are

saving a new file, its directory entry will be written to the space that used to be occupied by the file you deleted.

Undeleting files is tricky, but managable if the above considerations, as well as those pointed out in Chapter 7, are kept in mind.

Appendix E

DISK POTPOURRI

There are several interesting locations on a disk that can be modified to achieve results that the user may desire.

This is NOT a comprehensive list. With a little bit of time and a minimum amount of trouble, you can probably find many more yourself.

TRACK	SECTOR	BYTE	USE
1	7	$84	Beginning of DOS command table (text)
1	8	$74	Beginning of DOS error message table (text)
1	9	$75	Name of user's "Hello" program stored here
1	9	$B8	Name of program to be run, if "Hello" program is an Applesoft® program and that language is not resident in memory.
1	$C	$A4	Number of lines to print before waiting for key press when printing disk directory (when user gives "Catalog" command)
2	2	$AF	Text "DISK VOLUME" stored (backward). Used when "Catalog" command is issued.
$11	Ø	ALL	VTOC
$11	1	ALL	Last directory sector
$11	$F	ALL	First directory sector

Appendix F

APPLE ASCII CHART

Apple computer, as we all know by now, does not use lower-case letters, so the normal ASCII code was modified slightly to allow certain special functions on the Apple display. These functions are the FLASH and INVERSE commands of BASIC. Also, certain disk text storage of commands and information requires that the high bit be "on" in order that the information is translated correctly and used properly. For example, in the DOS command table (text), the DOS commands are in normal ASCII, except for the last character of each command, which has the high bit set. This way, DOS knows when one command ends and the next one starts.

A chart follows showing ASCII codes with numeric values and keyboard values. The numeric values are given in decimal, hex, enhanced decimal, and enhanced hex. In this instance, "enhanced" signifies that the high bit is on in the value given.

These are ASCII values for the Apple, so the lower case character values have been left off.

Appendix G

ASCII CODE TABLE

DEC	HEX	ENH DEC	ENH HEX	CHAR	KEYBOARD
Ø	ØØ	128	8Ø	NULL	ctrl-@
1	Ø1	129	81	SOH	ctrl-A
2	Ø2	13Ø	82	STX	ctrl-B
3	Ø3	131	83	ETX	ctrl-C
4	Ø4	132	84	ET	ctrl-D
5	Ø5	133	85	ENQ	ctrl-E
6	Ø6	134	86	ACK	ctrl-F
7	Ø7	135	87	BEL	ctrl-G
8	Ø8	136	88	BS	ctrl-H
9	Ø9	137	89	HT	ctrl-I
1Ø	ØA	138	8A	LF	ctrl-J
11	ØB	139	8B	VT	ctrl-K
12	ØC	14Ø	8C	FF	ctrl-L
13	ØD	141	8D	CR	ctrl-M
14	ØE	142	8E	SO	ctrl-N
15	ØF	143	8F	SI	ctrl-O
16	1Ø	144	9Ø	DLE	ctrl-P
17	11	145	91	DC1	ctrl-Q
18	12	146	92	DC2	ctrl-R
19	13	147	93	DC3	ctrl-S
2Ø	14	148	94	DC4	ctrl-T
21	15	149	95	NAK	ctrl-U
22	16	15Ø	96	SYN	ctrl-V
23	17	151	97	ETB	ctrl-W
24	18	152	98	CAN	ctrl-X
25	19	153	99	EM	ctrl-Y
26	1A	154	9A	SUB	ctrl-Z
27	1B	155	9B	ESCAPE	ESC
28	1C	156	9C	FS	n/a
29	1D	157	9D	GS	ctrl-shift-M
3Ø	1E	158	9E	RS	ctrl-^
31	1F	159	9F	US	n/a
32	2Ø	16Ø	AØ	SPACE	SPACE

DEC	HEX	ENH DEC	ENH HEX	CHAR	KEYBOARD
33	21	161	A1	!	!
34	22	162	A2	"	"
35	23	163	A3	#	#
36	24	164	A4	$	$
37	25	165	A5	%	%
38	26	166	A6	&	&
39	27	167	A7	'	'
40	28	168	A8	(	(
41	29	169	A9	)	)
42	2A	170	AA	*	*
43	2B	171	AB	+	+
44	2C	172	AC	,	,
45	2D	173	AD	-	-
46	2E	174	AE	.	.
47	2F	175	AF	/	/
48	30	176	B0	0	0
49	31	177	B1	1	1
50	32	178	B2	2	2
51	33	179	B3	3	3
52	34	180	B4	4	4
53	35	181	B5	5	5
54	36	182	B6	6	6
55	37	183	B7	7	7
56	38	184	B8	8	8
57	39	185	B9	9	9
58	3A	186	BA	:	:
59	3B	187	BB	;	;
60	3C	188	BC	<	<
61	3D	189	BD	=	=
62	3E	190	BE	>	>
63	3F	191	BF	?	?
64	40	192	C0	@	@
65	41	193	C1	A	A
66	42	194	C2	B	B
67	43	195	C3	C	C
68	44	196	C4	D	D
69	45	197	C5	E	E

ASCII CODE TABLE CONT.

DEC	HEX	ENH DEC	ENH HEX	CHAR	KEYBOARD
70	46	198	C6	F	F
71	47	199	C7	G	G
72	48	200	C8	H	H
73	49	201	C9	I	I
74	4A	202	CA	J	J
75	4B	203	CB	K	K
76	4C	204	CC	L	L
77	4D	205	CD	M	M
78	4E	206	CE	N	N
79	4F	207	CF	O	O
80	50	208	D0	P	P
81	51	209	D1	Q	Q
82	52	210	D2	R	R
83	53	211	D3	S	S
84	54	212	D4	T	T
85	55	213	D5	U	U
86	56	214	D6	V	V
87	57	215	D7	W	W
88	58	216	D8	X	X
89	59	217	D9	Y	Y
90	5A	218	DA	Z	Z
91	5B	219	DB	[	n/a
92	5C	220	DC	\	n/a
93	5D	221	DD	]	shift-M
94	5E	222	DE	^	^
95	5F	223	DF	_	n/a

INDEX

A

Applesoft® , 4, 15
ASCII, 22-23, 26, 44, 57-60

B

Back-up copies, 3, 7
Base convert, 39-40
BASIC, 11, 44
Binary, 15-16, 39, 44
Booting, 3-4, 9, 11, 44, 48
Byte, 22, 28, 44

C

Catalog, 9, 16, 44, 56
Create exec files, 33-34
CRT, 3, 23, 26, 44
Ctrl-C, 41
Ctrl-P, 37
Ctrl-S, 27

D

Decimal, 17, 22-24, 39, 44, 57
Delete, 16, 19, 51, 54-55
Directory, 19, 44, 47, 49-51
Disk
  copy, 31
  editor, 21-29
    command descriptions, 21-29
    command line, 21
    disassembler, 27
    display modes, 23-27
    display page, 23, 25
    DOS 3.3/3.2 Editor differences, 20
    edit types, 22
    input, 28
    status, 28-29
  map, 13-14
DOS versions, 7

E

Edit exec files, 35-36
Exec files, 33-36, 45

F

Files
  code, 16
  length, 16
  sectors used, 17
  text, 16
  types, 16, 51
  undeleted, 19
Flashing, 23, 57
Format blank disk, 9, 11

G

Getting started, 3-4

H

Hexadecimal, 22, 24-25, 40,45, 57

I

Improved directory, 16
Init, 9
Inverse, 17, 23, 52

L

Language card, 4

M

Main program menu, 3
Map, disk, 13
Master Create, 11

Mnemonics, 26
Monitor, 19, 45

O

Octal, 40

P

Pirates, 31
Printer, 27, 37
Program menu, 3
Purpose of copy programs, 31

R

Reset, 37, 40

Q

Quit, 34, 36, 40

S

Safety precautions, 7
Sector listings, 17
Sectors
  definition, 46
  free, 14
  in-use, 13
  pointer, 22-23

T

Text dump, 37-38
Toggle, 23, 27, 46
Track
  0, 9
  1, 9, 11
  2, 9, 11
  17, 9
  allocation order, 48
  bit map, 19, 48, 54
  borders, 23
  copy, 31
  definition, 46
  /sector list, 46, 51

U

Undeleted files, 19
User notes, 43

V

VTOC, 9, 46-48, 56

Allen Wyatt has been actively involved with the microcomputer industry for four years and is currently a program consultant and editor with Advanced Operating Systems in Michigan City, Indiana. Mr. Wyatt also acts as a computer consultant to several private businesses in the northern Indiana area.

He has written several commercial software packages for the Apple computer. The broad range of programs runs the gamut from small system data bases to games and utilities. APPLE-AIDS is a fine example of the latter category.

Besides being a computer author, Al is a devoted family man and active church member. He uses his personal Apple computer to assist him in all of these areas. At home, his family spends many hours using the computer every day.

ITT

HOWARD W. SAMS & CO., INC.
Developed by Advanced Operating Systems
4300 W. 62nd Street
Indianapolis, Indiana 46268
(317)298-5566

26066

ISBN: 0-672-26066-2